

## Interactive Solving of Vehicle Routing and Scheduling Problems: Basic Concepts and Qualification of Tabu Search Approaches

Herbert Kopfer

University of Bremen, Chair of Logistics  
[kopfer@logistik.uni-bremen.de](mailto:kopfer@logistik.uni-bremen.de)

Jörn Schönberger

University of Bremen, Chair of Logistics  
[sberger@logistik.uni-bremen.de](mailto:sberger@logistik.uni-bremen.de)

### Abstract

*This paper introduces a framework for the interactive solving of optimization problems and presents an interactive tabu search algorithm for the pick-up-and-delivery problem with time windows (PDPTW). The framework for the development of interactive algorithms is based on the idea of activating and deactivating constraints in the model of the problem under consideration. We start with an introduction to interactive problem solving and suggest an algorithmic framework. Next we focus to vehicle routing and scheduling applications. There we discuss the demand for interactive approaches, since established local search methods fail to produce reasonable solutions. Our idea is to support them by interactive manipulations. Therefore, we present a recent tabu search algorithm for the PDPTW. We demonstrate several possibilities for human interactions by means of an additional tabu list.*

### 1. Introduction

The solution of problems arising from the assignment and scheduling of transportation resources is subject of a hardly countable number of research projects. Different approaches are described in corresponding publications. The application of exact mathematical programming based methods is investigated as well as the qualification of heuristic search paradigms like problem-oriented tree-search approaches, decomposition techniques or methods inspired by nature.

Due to the complexity of the observed problems mathematical programming approaches (e.g. integer linear programming) fail to produce reasonable solutions. This contributes to the success of heuristic approaches. Meta-heuristics which support local search approaches are proposed and prove their applicability. The concepts of these search paradigms directly lead

to problems related to pre-emptive sub-optimal convergence. To work against this trouble we propose the application of interactive problem solvers which join the advantages of local search based automatic problem solvers with human inspiration and pattern recognition in order to improve the quality of the problem solutions and to enlarge the set of solvable problems.

Subject of this investigation is the presentation of a concept for interactive solving processes and the implementation of this concept in order to improve the quality of local search approaches for difficult highly constrained optimization problems using the skills of an external human decision maker. We apply the derived methods to problems out of the field of vehicle routing and scheduling.

We start our investigation in chapter 2 with the presentation of the advantages and possibilities of the process of interactive problem solving. Then we present the general framework of step-by-step problem solving and propose a concept for the realization of this framework. We consider the strengths of automatic and human problem-solvers and discuss a metaphor system for the communication between machine and human planner. At the end of chapter 2 we outline situations in which human actions together with machine power seem to be powerful. In chapter 3 we focus on interactive local search algorithms and their application to vehicle routing and scheduling problems. We show the limits of batch oriented local search algorithms, demonstrate the demand for interaction, and discuss a metaphor system for vehicle routing and scheduling problems. Chapter 4 presents the main ideas and results of an interactive tabu search algorithm for the pick-up-and-delivery problem with time windows. The presentation of the algorithm concentrates on the definition of the neighborhood and the possibilities and effects of human interventions. Chapter 5 terminates our paper with some conclusions and suggestions for future research.

## 2. Interactive Problem Solving

### 2.1. From batch processing to interactive processing

To get an automatically generated solution for an optimization problem we usually have to proceed as described in the following. First we analyze and formalize the problem under consideration. Then we choose an automatic solver and post the problem to the solver, i.e. describe the problem exactly in a representation which is suitable to the chosen solver and parameterize the optimization tool. Then the solving process starts and the underlying algorithm continues its search until it reaches a predefined termination criterion. The process of finding the (one or more) solutions is not manipulable after the algorithm has been started. This concept of problem solution is called “batch processing” (cf. [12]) and the corresponding algorithm is called an automatic problem-solver. The complete and irreversible definition of the problem is one distinguishing mark of batch processing. It is not possible to change the problem because there is no chance to propagate new information to the once started solving process. Additionally, we cannot influence the search process and its direction after the automatic search has once started.

Despite of the indisputable success and effectiveness of batch processing, its application is connected with several deficiencies:

1. In many real world problem instances the defining data are not completely known when the solving process has to be started or some of the data can change after the start. In this situation we cannot expect to get reasonable solutions without an information update.
2. Due to the complexity of many real world problems the search algorithm is not able to find a solution for the problem at all or cannot find a solution within an acceptable amount of time. During the run-time of the algorithm a human planner might be able to give hints for successful search ideas, but the algorithm cannot follow these hints because the underlying search paradigm doesn't fit to the view of the user. In addition there is no chance for the human user to implement his hints in the generated solution until the algorithm terminates.
3. In some cases it is not possible to code all required real world restrictions in a formal problem description which is the foundation of the automatic search process. The ignorance of real world requirements in the

solution finding process may lead to useless solutions, which can't be implemented.

In any of the above three cases the user has to take the solution which has been generated by the algorithm as a basis for further modifications and adaptations. The process of modification and adaptation is done by the user without any support of the search algorithm.

The deficiencies of batch processing problem solving in vehicle routing and scheduling are commonly known to the researchers and some approaches have been tested to overcome them. Most effort is spent to fill out the gap of the unknown future input data: one tries to anticipate these data using e.g. stochastic estimations and then optimizes an expectation (cf. [5], [17]). Another approach consists of the embedding of automatic problem-solvers into Decision Support Systems ([6], [23]), however these systems often play the role of a data management system. To guide a search process through the search space more effectively and efficiently, one tries to use dynamic or self adaptive solver parameterizations. But how can we put information into the search process which can't be coded in advance in formal phrases? We need a possibility to provide information to the search process at run-time. This is provided by the so called “interactive processing” (cf. [12]). In this case, the problem at hand is posted to the algorithm of the solver piece by piece. These pieces are called “partial problems”. Modifications which are done by the human solver are recognized and the interactive algorithm continues its optimization task under consideration of the additional information and the problem-specific hints given by the user.

### 2.2. Step-by-Step Problem Solving

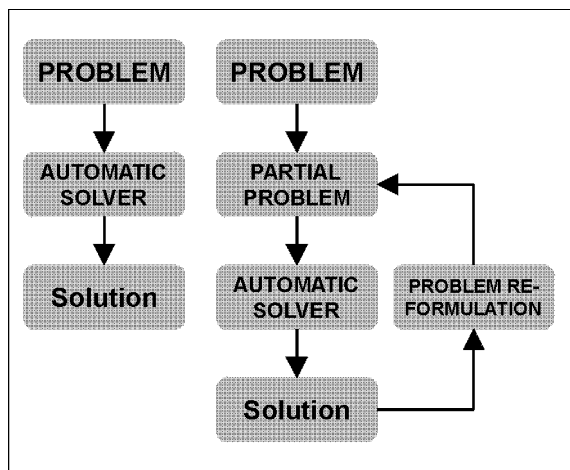
To get a solution in an interactive way we follow a general framework which differs from batch processing approaches. At time  $t=0$ , we start by formulating the initial partial problem using a subset of the problem data. This partial problem is posted to the solver which establishes the solution process. It stops as soon as a termination criterion is reached. Then we can replenish or modify the (formal) problem description and reject or accept the decisions which have been made until the current time  $t=t_1$ . The replenishment and the performed modifications lead to a reformulation of the problem description. After the completion of the manual interventions, for instance by a human

planner, we recall the solver to solve the new problem and so on.

A solver which is started successively in this way is called an “interactive problem solver” and the successive process of terminations and recalls will be denoted as “interactive problem solving”. So an interactive problem solver represents a cyclic batch processing (cf. Fig. 1). Finally, we define the solution of a partial problem as “preliminary solution”.

We remark that the solution, found at the end of an interactive process cannot be guaranteed to be optimal and that the manual interventions cause an additional effort, which should not be underestimated. But if there is no chance to use a batch processing solver, one should accept the generated sequence of preliminary solutions and try to implement it.

To apply the interactive solving process to a problem, we have to divide the defining information of the considered problem to generate a sequence of partial problems. This breakdown has to be done very carefully since one does not want to create a sequence of easy to solve but independent sub-problems whose solutions are not able to be merged to a solution of the real world problem under consideration. So a reasonable amount of problem knowledge is needed to derive an appropriate problem sequence. Often, an obvious way of sequencing is inherent in the problem.



**Fig. 1: schema of a batch-oriented solver (left) and successively interactive processing (right).**

For instance, for dynamic vehicle routing and scheduling problems (cf. [18]), problem data changes over time: new customer requests, request cancellations, vehicle breakdowns or traffic jams appear and have to

be integrated into a planning process to find an implementable schedule for the transportation resources. For this kind of problems, a partial problem at time *t* is canonically defined by the data known up to time *t*. Following this definition we construct an interactive solver whose underlying model is reformulated every time new data will be available and which is started straight afterwards.

### 2.3. A concept for interactive problem solving

We have to find a way to construct a useful and effective sequence of partial problems. In this subsection an algorithmic concept for an interactive problem solver is introduced. The concept is founded in posting a relaxed problem to the local search solver and let it generate a solution for this problem. It's understood, that the returned solution doesn't respect all required constraints because they haven't been coded into the model and so the solver doesn't know them. The main idea is to try to add successively the missing constraints to the problem in a way that the solver gets a slightly reconfigured problem. In addition the removal of selected constraints is possible. Fig. 2 shows this approach using pseudo code.

In this investigation, the decision which constraints are to be added or removed is left to an external decision maker, especially a human planner. The same holds for the decision whether the solutions which have been found so far are of a quality that allows the termination of the improvement process.

The implementation of an interactive problem solver isn't an easy task if we take into account the following targets of an interactive approach. Firstly, we have to find a way to show to the human planner the solution found up to now. Secondly, such a system should present some alternative ways to proceed. Finally, the system should support the decision for one of the available alternatives and show to the human planner the consequences of his actions. The last two wishes require powerful procedures, which incorporate the formal problem representation.

The concept introduced above is supported by the significant advantage of the presence of a current and nearby feasible solution throughout the whole search process. If the system saves the feasible solutions once found, the property of reversibility can be reached. This means that the user can try more freely new concepts because he doesn't have to take care that he loose a feasible solution. A feasible solution of a preliminary iteration can be taken from a stack whenever

the feasibility has been destroyed by manual interventions.

```

I=0;
PI=relaxed(P);
SI=solve(PI);
while(quality(SI) is unacceptable)
{
  RI+1=set of constraints to remove from PI;
  AI+1=set of constraints to add to PI;
  PI+1=reconfigure_problem(PI,RI+1,AI+1);
  I=I+1;
  SI=solve(PI);
}
terminate();

```

**Fig. 2: Pseudo code of the interactive concept.**

#### 2.4. Combining the strengths of automatic and human problem-solvers

Interactive problem solvers achieve their power by the consolidation and effective cooperation of two specialized actors (cf. [4]): first there is the human planner who uses historical problem knowledge and intuition for the creative solving of unknown or new problems. He is able to recognize patterns, can appreciate the consequences if restrictions should be temporarily relaxed and finally he can estimate the value of the balance between conflicting objectives. On the other side there is the machine with its own indisputable advantages in handling large data sets, processing extensive calculations and the exact and quick visualization of large amounts of data and their dependencies.

We have to distinguish between the characteristics and goals of dynamic and interactive problem solving. A dynamic problem consists of a sequence of static problems. For each of these static problems we have to find a feasible solution, which has to be implemented immediately. Using an interactive approach we perform in principle the same tasks but our real goal is not to find a solution for each partial problem. We only have to find a reasonable solution of the overall problem and make use of the solutions of the partial problems. We combine and adapt the preliminary solutions using the specific strengths of the human and the automatic problem solver.

#### 2.5. Communication between planner and machine

The cooperation between machine and human planner needs an extensive data interchange between these two actors. This requires the matching of the human's problem representation and the internal formal model. The man-machine-interface of an interactive problem solver has to overcome two main difficulties:

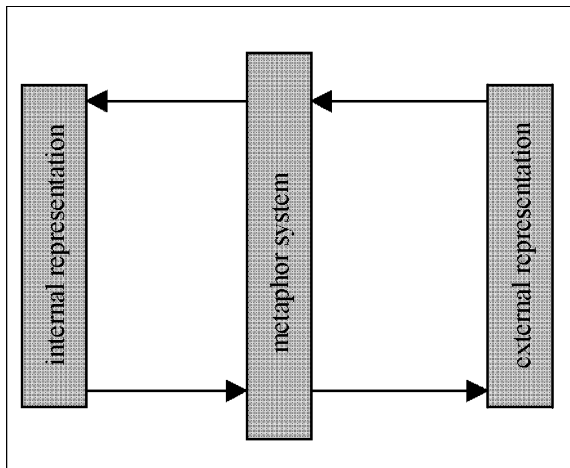
- The graphical user interface (GUI) has to represent the preliminary results reached up to now and to provide an input mechanism for the user's problem reconfiguration.
- To support the automatic solving process we have to establish arrangements, which control the additional input that should lead to a reformulation of the problem to solve. The main reason is to prevent the user from the propagation of inconsistent information that does not improve the quality of the solution and the search for it.

One main challenge in the development of a system for interactive problem solving is the matching of the human problem view, in most of the cases graphically inspired, and the formal algorithm dependent representation of the problem. There are several ideas to realize the communication between human planner and the machine, like command languages, menus and forms, but the most important concept is the use of metaphors and their direct manipulation with an input device (mouse, trackball, light pen, etc.).

We define a metaphor as the representation of one or more problem components (including their interdependencies) which is understandable intuitively by the human planner. Due to the graphical human problem understanding and with respect to the graphic systems available, there are a lot of possibilities to define graphic metaphors. If the metaphor represents more than one problem component, we have to assure that the interdependencies of the components are taken into account and that they can't be destroyed by human interactions.

In addition, the realization of a (graphical) metaphor has to consider ergonomic aspects in order to support the user acceptance of the system and to allow powerful interventions by the user. Naturally, the metaphor-system should reject error-inherent user inputs to protect the solutions found up to now and to prevent unrealizable solutions. Thus we can summarize the functions of the metaphor-system:

- The metaphor-system has to convert the internal representation to the external representation and vice versa (cf. fig. 2).
- The metaphor-system has to present reasonable problem components and the progress of the search for a solution.
- The metaphor receives and tests the external inputs.



**Fig. 3: metaphor-system as mediator between machine and human**

### 2.6. Problem domains of interactive systems

After the introduction of a concept for interactive problem solving and the discussion of the strengths of human and automatic problem solvers we try to describe situations in which the interactive approach is worth to be taken into account. There are two main areas of applications of interactive problem solving.

**Incompletely defined problems.** We call a problem incompletely defined if not all possible decisions and restrictions of the corresponding real world problem can be mapped in a model. There are several reasons for incomplete problem descriptions: special features can't be mapped in the chosen formal mathematical model, the consideration of special features leads to an unacceptable complexity explosion of the model, features become known after the model building process is terminated, the importance of a feature changes. If we post a model to a problem solver, the solver takes into account only those features which are coded in the model. Conversely, a feature which is not adequate coded in the model is ignored by the solver and can't be considered during the solving process of the real world problem. Later on, we consider some situations

out of the field of vehicle routing and scheduling, which may lead to incompletely defined problems.

**Badly structured problems.** The way we represent a real world problem as a formal model defines the structure of the problem (more exactly: of the model which represents the problem at hand). The solver only recognizes the formal structure of the posted model. If the structure provokes the search algorithm to scan preferentially the solution space in an inefficient way, we call the model badly structured. Treating a badly structured model, the search process wears off in unspectacular regions of the search space without identifying and collecting new information. Unfortunately, in most cases we are not able to recognize directly if a presented formal model is well or badly structured. Using an interactive problem solver, we can monitor the search and recognize when the search algorithm gets inefficient or runs into a deadlock situation. In these cases, we hope to be able to influence the search process and tear it for instance out of the deadlock situation.

In the following section we investigate, how we can improve local search approaches by integrating them in the proposed concept for interactive problem solving.

### 3. Interactive Local Search for Vehicle Routing and Scheduling Problems

Vehicle routing and scheduling covers all problems of the assignment of transportation requests to the available transportation resources and the sequencing and scheduling of the resulting processes. We refer to [2], [3], [16] and [9] for survey articles.

The process of problem solving begins with a formalization and the derivation of a suitable mathematical programming oriented optimization model (often formulated as linear program with integer constraints). Subsequently, this model is posted to an available solver. On the other hand [13] and [10] identify real-world-problem-components, which can't be integrated into a reasonable formal model, so that interactive manipulations are dedicated to support the process of solving the problem. Examples for such components are given below in subsection 3.2.

Recent approaches use the concept of local search to generate or to improve solutions of vehicle routing and scheduling problems. The mapping of almost all required real world properties leads to highly constrained optimization problems. For such problems even the hybridization of local search with powerful

search paradigms like simulated annealing and tabu search often fails to produce solutions of reasonable quality. In the following subsections we summarize the limits of the local search approaches and motivate the application of interactive strategies in cooperation with local search. Then we focus to real world situations in the field of vehicle routing and scheduling which tend to irritate local search algorithms. We describe how to overcome these situations with the help of interactive interventions. To the best of our knowledge there is no interactive local search approach described following the concept introduced in this paper, although the idea of improving the solution quality using human resources and machine power together is known for decades (cf. [22]).

### 3.1. Limits of batch oriented local search procedures

Before we present our concept of using external knowledge in local search approaches, we outline the idea of local search procedures. A local search algorithm picks up an existing (feasible or infeasible) solution of the problem at hand, performs a predefined manipulation (one move) and gets back a new solution. The main idea is to find successively solutions of increasing quality (measured by the chosen objective function). More formal, the “neighbourhood”  $N(p)$  of a solution  $p$  is defined as the set of all (feasible or nearly feasible) solutions of the problem which can be reached by applying one move to the solution  $p$ . A local search algorithm picks up a solution  $p$  and chooses one element  $p'$  of  $N(p)$  to be the next current solution. The rules of choosing a new current solution depend upon the algorithm in use. If we assume

- that the elements of a neighbourhood are highly correlated to the current solution  $p$  (correlation),
- that only a small part of all possible moves destroy the feasibility (feasibility),
- that there is a good chance to find a solution of higher quality in the neighbourhood (improvement),
- that there is a finite sequence of moves which transforms the initial solution to a solution with reasonable quality (connectivity),

the application of local search seems to be powerful enough to get a significant overall quality increasing by performing a number of short steps [cf. 14]. Especially the presence of the connectivity property is significant and we conjecture, that this property drifts away if the number of incorporated constraints increases. This seems to be an explication, why the con-

cept of local search often leads to solutions of unreasonable quality for highly constrained problems. Especially for this kind of problems an interactive approach seems to be promising.

### 3.2. Demand for interaction

The following situations of vehicle routing and scheduling problems are difficult to handle by an automatic problem solver (i.e.: a batch processing solver) and should be handled by the support of a human planner:

It is often necessary to prefer a temporarily valid objective, which is conflicting to the global objective coded in the model. In such situations the target of the search algorithm differs from the currently most important actual (maybe myopic) target. Without any external help the automatic search ignores this myopic strategy. In this context, we refer to situations, in which few but very important requests are late and we have to deliver these requests as soon as possible.

In the same manner a human planner can post a temporary restriction which concerns a transportation resource or the underlying transportation network. This becomes important in the presence of short or middle term traffic jams or if a truck breaks down but will be repaired after a short period or if special equipment (cooling, crane) turns out for a short period.

In many situations there are special relationships between a customer of a transportation company and the driver, which serves this customer. This relationship is founded by satisfying work, which is the result of the individual contact between the driver and the employee of the customer. There is a very high and fruitful degree of trust or knowledge of the local conditions related to this customer. This cooperation should not be destroyed, for instance by the assignment of changing drivers.

A well parameterized vehicle routing and scheduling system tries to reach more than a single objective. In addition to the cost-based view, service-oriented and ecologically inspired targets have to be taken into account. To prevent the domination or the sinking of single aspects in the target-system, a human planner must control the balance between these aspects.

There are differences in the importance of the customers to the transportation service provider, e.g. “VIP-customers” whose contribution to the success of the carrier is very high. These customers are worth to make special decision in the planning process, for instance if there is a questioned express last-minute request, which leads to temporarily objective-

reconfigurations as mentioned above. The implementation of these “VIP”-requests effects the necessity of rescheduling-decisions. This effect is very delicate, if the arrival times have already been announced to other customers. In these cases, a human planner should find a balance between the last-minute-service and the need of rescheduling.

A human planner can recognize unstable solutions: often the quality of a schedule depends upon a very small number of key assignments. If these assignments become obsolete (due to some breakdowns or new customer requests) the whole schedule loses quality or becomes obsolete at all. To avoid these unstable situations a human planner has to recognize them and initiate decisions which support the permanent good quality of the determined schedule and balance the costs of these decisions. This kind of sensitivity analysis should be left to a human planner (cf. [10]).

Finally, at the start of the optimization process, no one knows if there exists a solution to the given problem. In the case that no reasonable solution is found, only a human planner can decide which constraints can be relaxed without losing the ability of implementing the schedule (emergency strategies).

Since we want to prevent from handling large scale and complex models and since we want to consider human skills within the search process, to our opinion, it is worth to investigate interactive approaches. However, in special situations, specialized fine-tuned solvers can also be appropriate but they do not have the flexibility of interactive approaches.

### 3.3. Metaphor system

As already remarked above we have to choose and design carefully a metaphor system which works as an interface between machine and human planner. The realization of this interface should be done with respect to the following aspects:

1. The metaphor system receives the user input. It should be able to realize obvious faults and inconsistencies before the input is transferred into the formal problem description. In case of vehicle routing and scheduling problems the user should be instructed which available transportation resources are left and the system should warn the user if he tries to implement resource-critical assignments.
2. The metaphor system should represent the results found by the optimization algorithm so far. This means it can immediately be seen which requests have already been assigned and which requests are still

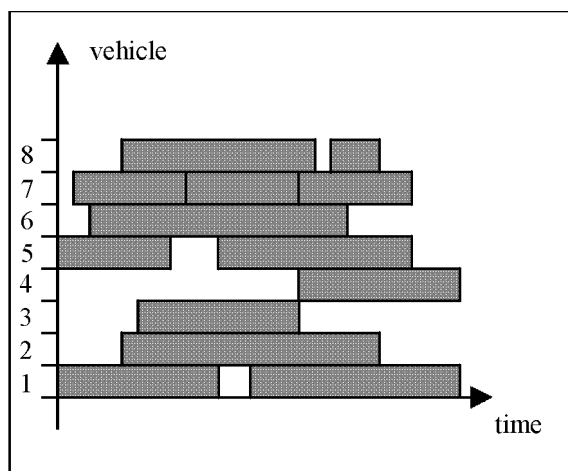
open. The most important performance indicators should be updated after each optimization task and after each manual manipulation by the user.

3. The metaphor system has to support the user’s way to identify important problem components and to recognize promising hints for good solutions. As an example, we think that the spatial distribution of requests indicates which requests should be tried to combine in order to save transportation resources. On the other way time windows should be represented in a suitable way.

As a result we obtain the need of a problem representation with five dimensions:

- amounts of contribution,
- importance of special customer requests,
- resource demand of the requests,
- time dimension and,
- spatial distribution,

Clearly, there’s no way to show a reasonable representation of these aspects at the same time. We propose to create a metaphor system which shows the user a selected collection of these aspects. In addition, the decision which aspects are shown should be left to the human planner, who should decide this question dynamically to adopt his own demand.

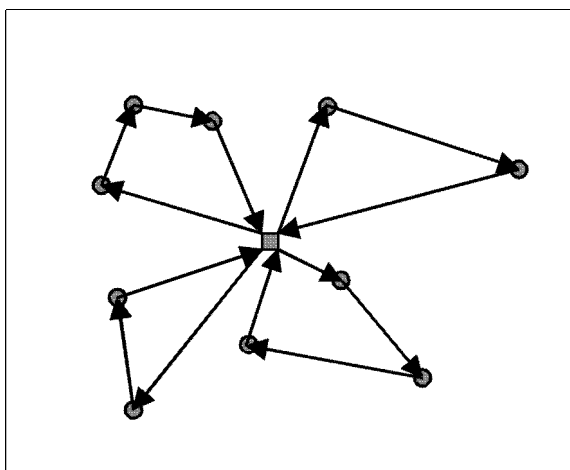


**Fig. 4: A Gantt-diagram for schedule Representation**

To realize the metaphor system there are several tools. The presentation of a list of important performance indicators is easy to implement, nevertheless it is very important. We think that a list-oriented representation is appropriate for the amounts of contribution, the importance of the customer requests, and the re-

source demands of the requests. With respect to previously defined weights, these lists can be sorted so that the order represents the importance of assignment.

To visualize the time a transportation resource spends to fulfil a customer request, we propose the usage of a Gantt-diagram (cf. Fig. 4). Using this representation the identification of slack times is easy, but on the other hand there's no way to represent the spatial distribution of the customer requests in order to find hints for cluster-building.



**Fig. 5: Visualization of a schedule by a graph with edges and knots.**

This observation predicts the graph-based problem representation to be more valuable to vehicle routing and scheduling problems. We state that there is no possibility to represent time windows and related problem components in a graph-based representation.

From our point of view there is not the one and only metaphor system for vehicle routing problems, there should rather be the possibility to switch between different representations and to adapt the representation to the actual situation.

#### 4. A Tabu Search Algorithm for the Pickup-and-Delivery Problem

In this section, we focus on the use of Tabu search algorithms in an interactive solver system. We start with a short outline of the Tabu search paradigm. Afterwards we present an interactive Tabu search algorithm and demonstrate how Tabu search implementations can be used to support interactive manipulations in the process of problem solving.

#### 4.1. The Idea of Tabu Search

Tabu search is a meta-strategy to guide local search procedures. The main goal of the strategy is to prevent the search process from a cycling in local optima. To reach this, special moves are forbidden (set "tabu"). Forbidden moves or the characteristics of forbidden moves are collected in a so called tabu list. The identified moves are forbidden because they reverse previously performed moves or because they would guide the search along a trajectory, which has been explored just before. This leads to modified neighborhoods  $N(H,p)$  because all elements of  $N(p)$  which are reached by applying a move of the tabu list  $H$  to  $p$  are removed from the neighborhood in order to enforce the remaining elements. In addition to the short term memory which aims to prevent cycles and is represented by the tabu list, tabu search uses intermediate and longer term considerations for the intensification and diversification of the search process. Intensification strategies force the thorough investigation of promising regions of the search space and are based on the short, intermediate and long term memory of the search process. In the short term this consists of incorporating attributes of solutions receiving highest evaluations, while in the intermediate to long term it consists of incorporating attributes of solutions from selected elitist subsets (implicitly focussing the search in sub-regions defined relative to these subsets). Diversification strategies instead seek to generate solutions that embody compositions of attributes different from those encountered previously during the search. These two types of strategies counterbalance and reinforce each other in several ways (cf. [8]).

There are several strategies for the implementation and the management of tabu lists. For our further investigations, only the concept and the implications of the use of tabu lists are important. A detailed introduction to tabu search can be found in [8].

Tabu search approaches have been successfully applied to the field of vehicle routing and scheduling in various publications (cf. [15], [7], [10], [1], [21]). In [10], a tabu search algorithm for the well known pickup-and-delivery-problem (PDPTW) (cf. [20]) is developed and presented. This algorithm is tested and integrated in the software system INTU (cf. [11]) which provides a GUI and drag-and-drop-manipulations to create problem instances and to influence the search process by manipulating the tabu list.

## 4.2. Interactions in INTU

In this subsection we describe the definition of the neighborhood and the use of tabu lists to integrate the interactive components within the INTU system.

The main idea of INTU is the successive description and solving of the PDPTW by using an additional tabu list which is generated and manipulated by interventions of the human user. INTU starts with an interactively generated description of an initial problem and solves this problem with a parallel-best-insertion based algorithm. This initial solution is improved by a tabu search based local search until a predefined amount of processing time is reached. Then the best solution found is presented using the GUI. This solution is the first starting-point for further interactive processing.

The algorithm uses a combined neighborhood consisting of a delete-insert-neighborhood and an exchange-neighborhood. Using the delete-insert-neighborhood, a move of the algorithm is performed by the deletion of a customer-request in a tour of the current solution and by finding the best way to insert the deleted request in another tour. In the exchange-neighborhood a single move interchanges the assignment of two customer requests to their corresponding tours. Each of the interchanged customer requests is inserted in the former tour of its changing partner using a heuristic scheduler. A similar combined neighborhood is used in [15]. The main advantage of the delete-insert-neighborhood is the possibility to reduce the number of used tours, and the advantage of the exchange-neighborhood is its high connectivity.

Benchmarks have shown that the developed tabu search algorithm for the PDPTW produces good computational results, when it is used without any human intervention (cf. [10]).

INTU provides three kinds of manipulations:

The problem can be reconfigured by adding or deleting customer requests.

- (1) The set of available transportation resources can be modified.
- (2) The solution received so far can be modified by fixing or changing solution parts to get instructions to the solver in order to improve the solution quality in a subsequent solution process.

Manipulations of type (1) can be used to handle and to solve dynamic PDPTWs. Additionally they can support the human planner when he is deciding whether he should accept or reject a new customer request. Using this facility, INTU is not only an optimization

system for the PDPTW but also a system supporting the previous decisions concerning the order management.

Manipulations of type (2) have been included because INTU is able to solve an important extension of the PDPTW, which allows the hiring of independent cooperating carriers for some customer requests. In this case the optimization algorithm of INTU tries to minimize the total transportation costs, consisting of the sum of the prime costs for the use of the own trucks and the freight costs for the employment of carriers.

Manipulations of type (3) can be used in order to fix one or several attributes of the solutions which are to be considered during the search process. These attributes cannot be changed by the search algorithm and are maintained in an additional tabu list, which results from the human interventions. This additional tabu list is called interactive tabu list, and the tabu list which is maintained by the algorithm itself is called original tabu list. The interactive tabu list cannot be altered by the algorithm. The user can for instance fix the assignment of a customer request to a specific tour or the position of a customer request within a tour in the interactive tabu list. Then the tabu search algorithm only seeks for solutions which correspond to this fixation. By this way the user can manipulate the algorithm in situations which need a human intervention as outlined in subsection 3.2.

## 4.3. Navigating in the search space using tabu lists

There are two facets we have to remember if we talk about the ability of the interactive tabu list to support the search for reasonable solutions: The preservation of destruction of parts of promising preliminary solutions and the enforcement of exploring something new.

The most intuitive way to use the possibility of interactions is the prevention of changes in selected parts of the solution found up to now. We can understand this aspect as a possibility of fixing parts of a preliminary solution in the process of re-optimization. Following this way we are able to prevent the destruction of solution properties which have been found up to now and which have been identified to be successful. With respect to the tabu search algorithm this means an intensification of the search process which has been triggered by the human user.

The second facet deals with the external knowledge which has gathered to the problem up to now. Since the solver cannot overrule the given decisions we force him to consider solutions with certain attributes. Following this way we can try to work against dominating or algorithm-dependent solution properties in order to reach an increasing solution quality by an exploration of new regions of the search space. With respect to the tabu search algorithm this means a diversification of the search process.

## 5. Conclusion

In this investigation we presented an idea to support the solving of difficult vehicle routing and scheduling problems. We introduced the concept of interactive problem solving to overcome the commonly known deficiencies of local search procedures. The development of an interactive Tabu search algorithm and its application to the PDPTW has been reported.

Future research should be spent on the methods for development of suitable frameworks which support the implementation of interactive components. Especially, the design of useful metaphor systems is a challenge. This includes the definition of an interface in order to integrate the powerful search procedures described in literature.

The evaluation of interactive systems is not as easy as the evaluation of a batch-oriented algorithm because the variety of manual interventions and the dependence on the specific user lead to incomparable results. It is our opinion, that the numerical evaluation should first be restricted to the underlying batch-oriented optimization routines because this component of an interactive system is very important to the success. The metaphor system used can't be evaluated in an objective way, because every user prefers different components and has its own feeling of working appropriately. So, the metaphor system and the computational results of the interactive solving process could only be evaluated in exhaustive field experiments.

## 6. Bibliography

- [1] P. Badeau, F. Guertin, M. Gendreau, J.-Y. Potvin and E. Taillard: A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows in: *Transportation Research* 5(2), 1997.
- [2] J. Bramel and David Simchi-Levi: *The Logic of Logistics*, Springer, New York, 1997.
- [3] T.G. Crainic and G. Laporte (eds.): *Fleet Management And Logistics*, Kluwer Academic Publishers, London, 1998.
- [4] G. Diruf: Probleme und Entwicklungstendenzen der computergestützten Tourenplanung in: *Zeitschrift für Planung*, Nr. 1, 1990.
- [5] M. Dror, G. Laporte and P. Trudeau: Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks in: *Transportation Science* Vol. 23(3), 1989.
- [6] P. Duchessi, S. Belardo and J.P. Seagle: Artificial Intelligence and the Management Science Practitioner: Knowledge Enhancements to a Decision Support System for Vehicle Routing in: *Interfaces* 18(2), 1988.
- [7] M. Gendreau, F. Guertin, J.-Y. Potvin and E. Taillard: Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching in: *Transportation Science* 33(4), 1999.
- [8] F. Glover and M. Laguna: *Tabu Search in: C.R. Reeves (ed.), Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, 1993.
- [9] B.L. Golden and A.A. Assad (eds.): *Vehicle Routing: Methods and Studies*, Elsevier, Amsterdam, 1988.
- [10] T. Greb: *Interaktive Tourenplanung mit Tabu Search*, Ph.D.-thesis, University of Bremen, 1998.
- [11] T. Greb: *INTU – A User Manual*, University of Bremen, 1998.
- [12] H.R. Hansen: *Wirtschaftsinformatik 1, 6. Ed.*, G. Fischer-Verlag, Stuttgart, 1992.
- [13] H. Kopfer: *Heuristische Suche in Operations Research und Künstlicher Intelligenz: Strategien des heuristischen Problemlösens und die Lösung betrieblicher Dispositionssprobleme*, Habilitation thesis, Freie Universität Berlin, 1989.
- [14] D. Mattfeld: *Evolutionary Search and the Job Shop*, Physica, 1996
- [15] W.P. Nanry and J.W. Barnes: Solving the pickup and delivery problem with time windows using reactive tabu search in: *Transportation Research Part B* 23, 2000.
- [16] J. de D. Ortúzar and L.G. Willumsen: *Modelling Transport*, Wiley, Chichester, 1994.
- [17] W.B. Powell, P. Jaillet and A. Odoni: *Stochastic and Dynamic Networks in: Handbook in Operations Research and Management Science: Networks*, Elsevier Science Publishers, 1995.
- [18] H.N. Psaraftis: Dynamic Vehicle Routing: Status and prospects in: *Annals of Operations Research* 61, 1996.
- [19] C.R. Reeves (ed.): *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, 1993.
- [20] M.W.P. Savelsbergh and M. Sol: The General Pickup and Delivery Problem in: *Transportation Science* 29(1), 1995.
- [21] E. Taillard, P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin: A tabu search heuristic for the vehicle routing problem with soft time windows in: *Transportation Science*, 31(2), 1997.
- [22] C.D.J. Waters: Interactive vehicle routing in: *Journal of the Operational Research Society* 35(9), 1984.
- [23] C.D.J. Waters: Expert Systems for Vehicle Scheduling in: *Journal of the Operational Research Society* 41(6), 1990.